Amendments to the Specification:

Please replace paragraph [0001] with the following amended paragraph:

[0001] This application claims priority to U.S. Provisional Application Serial No. 60/400,391 titled "JSM Protection," filed July 31, 2002, incorporated herein by reference. This application also claims priority to EPO Application No. 03291909.4, filed July 30, 2003 and entitled "Using IMPDEP2 For System Commands Related To Java Accelerator Hardware," incorporated herein by reference. This application also may contain subject matter that may relate to the following commonly assigned co-pending applications incorporated herein by reference: "System And Method To Automatically Stack And Unstack Java Local Variables," Serial No. [[ ]]10/632,228, filed July 31, 2003, ~~Attorney Docket No. TI-35422 (1962-05401)~~; "Memory Management Of Local Variables," Serial No. [[ ]]10/632,067, filed July 31, 2003, ~~Attorney Docket No. TI-35423 (1962-05402)~~; "Memory Management Of Local Variables Upon A Change Of Context," Serial No. [[ ]]10/632,076, filed July 31, 2003, ~~Attorney Docket No. TI-35424 (1962-05403)~~; "A Processor With A Split Stack," Serial No. [[ ]]10/632,079, filed July 31, 2003, ~~Attorney Docket No. TI-35425(1962-05404)~~; "Test With Immediate And Skip Processor Instruction," Serial No. [[ ]]10/632,214, filed July 31, 2003, ~~Attorney Docket No. TI-35427 (1962-05406)~~; "Test And Skip Processor Instruction Having At Least One Register Operand," Serial No. [[ ]]10/632,084, filed July 31, 2003, ~~Attorney Docket No. TI-35248 (1962-05407)~~; "Synchronizing Stack Storage," Serial No. [[ ]]10/631,422, filed July 31, 2003, ~~Attorney Docket No. TI-35429 (1962-05408)~~; "Methods And Apparatuses For Managing Memory," Serial No. [[ ]]10/631,252, filed July 31, 2003, ~~Attorney Docket No. TI-35430 (1962-05409)~~; "Write Back Policy For Memory," Serial No. [[ ]]10/631,185, filed July 31, 2003, ~~Attorney Docket No. TI-35431 (1962-05410)~~; "Methods And Apparatuses For Managing Memory," Serial No. [[ ]]10/631,205, filed July 31, 2003, ~~Attorney Docket No. TI-35432 (1962-05411)~~; "Mixed Stack-Based RISC Processor," Serial No. [[ ]]10/631,308, filed July 31, 2003, ~~Attorney Docket No. TI-35433 (1962-05412)~~; "Processor That Accommodates Multiple Instruction Sets And Multiple Decode

Modes," Serial No. [[ ]]10/631,246, filed July 31, 2003, ~~Attorney Docket No. TI-35434 (1962-05413)~~; "System To Dispatch Several Instructions On Available Hardware Resources," Serial No. [[ ]]10/631,585, filed July 31, 2003, ~~Attorney Docket No. TI-35444 (1962-05414)~~; "Micro-Sequence Execution In A Processor," Serial No. [[ ]]10/632,216, filed July 31, 2003, ~~Attorney Docket No. TI-35445 (1962-05415)~~; "Program Counter Adjustment Based On The Detection Of An Instruction Prefix," Serial No. [[ ]]10/632,222, filed July 31, 2003, ~~Attorney Docket No. TI-35452 (1962-05416)~~; "Reformat Logic To Translate Between A Virtual Address And A Compressed Physical Address," Serial No. [[ ]]10/632,215, filed July 31, 2003, ~~Attorney Docket No. TI-35460 (1962-05417)~~; "Synchronization Of Processor States," Serial No. [[ ]]10/632,024, filed July 31, 2003, ~~Attorney Docket No. TI-35461 (1962-05418)~~; "Conditional Garbage Based On Monitoring To Improve Real Time Performance," Serial No. [[ ]]10/631,195, filed July 31, 2003, ~~Attorney Docket No. TI-35485 (1962-05419)~~; "Inter-Processor Control," Serial No. [[ ]]10/631,120, filed July 31, 2003, ~~Attorney Docket No. TI-35486 (1962-05420)~~; "Cache Coherency In A Multi-Processor System," Serial No. [[ ]]10/632,229, filed July 31, 2003, ~~Attorney Docket No. TI-35637 (1962-05421)~~; "Concurrent Task Execution In A Multi-Processor, Single Operating System Environment," Serial No. [[ ]]10/632,077, filed July 31, 2003, ~~Attorney Docket No. TI-35638 (1962-05422)~~; and "A Multi-Processor Computing System Having A Java Stack Machine And A RISC-Based Processor," Serial No. [[ ]]10/631,939, filed July 31, 2003, Attorney Docket No. TI-35710 (1962-05423).

Please replace paragraph [0024] with the following amended paragraph:

[0024] The data storage 122 generally comprises data cache ("D-cache") 124 and data random access memory ("D-RAMset") 126. Reference may be made to co-pending applications U.S. Serial Nos. 09/591,537 filed June 9, 2000 (~~Attorney Docket No. TI-29884~~Now U.S. Pat. No. 6,826,652), 09/591,656 filed June 9, 2000 (~~Attorney Docket No. TI-29960~~Now U.S. Pat. No. 6,792,508), and 09/932,794 filed August 17, 2001 (~~Attorney Docket No. TI-31351~~Now U.S. Pat. No. 6,789,172), all of which are

incorporated herein by reference. The stack (excluding the micro-stack 146), arrays and non-critical data may be stored in the D-cache 124, while Java local variables, critical data and non-Java variables (e.g., C, C++) may be stored in D-RAM 126. The instruction storage 130 may comprise instruction RAM ("I-RAM") 132 and instruction cache ("I-cache") 134. The I-RAMset 132 may be used for "complex" micro-sequenced Bytecodes or other "micro-sequences or critical sequences of codes," as will be described below. The I-cache 134 may be used to store other types of Java Bytecode and mixed Java/CISA instructions.

Please replace paragraph [0027] with the following amended paragraph:

[0027] In a preferred embodiment, system commands (e.g., mask, unmask, micro-stack-clean, etc.) are accessible from the first and second instruction sets due to a common Bytecode in each instruction set. This Bytecode may be used to determine that the following instruction is a system ~~commands~~command. System commands may belong to one instruction set and preferably to the second instruction set. A system command, as described herein, is an instruction that executes a variety of system tasks essential for the execution of instructions. In some embodiments, the decode logic may be adapted to switch modes depending on an instruction and to which the instruction belongs. An instruction may be made up of one Bytecode, or the plurality of Bytecodes. However, the Bytecodes that make up the instruction in one instruction set may also be used in another instruction set but may not constitute the same instruction. As such, the decoder must be adapted to decode instructions based on the instruction set.

Please replace paragraph [0028] with the following amended paragraph:

[0028] Referring again to Figure 4, while the decode logic 152 is decoding Bytecode A, pre-decode logic 158 may pre-decode five Bytecodes following Bytecode A. In particular, the pre-decode logic may pre-decode Bytecodes defined by the implementation that indicates the mode the decode logic should operate in for at least a succeeding instruction. For example, if Bytecode A belongs to the first instruction set,

the pre-decode logic 152 may detect Bytecode B is a predetermined prefix which indicates at least one instruction from the second instruction set following the predetermined prefix corresponds to the system command. If Bytecode A belongs to the second instruction set, the pre-decode logic 152 may detect that Bytecode B is a predetermined prefix which indicates at least one system command follows, in which the system command belongs to the second instruction set. The predetermined prefix, common to both instruction ~~set~~sets, may be an implementation dependent Bytecode, such as a Java Impdep2 Bytecode. In a preferred embodiment, after detecting the predetermined prefix, the Bytecode succeeding the predetermined prefix is loaded into the decode logic 152. Referring to Figure 5, the decode logic 152, after decoding Bytecode A, may immediately decode Bytecode C, a system command when executing instruction in the first mode. Similarly, if the decode logic is operating in the second mode and the Java Impdep2 instruction is detected, the decode logic may remain in the second mode to decode the at least one system instruction.

Please replace paragraph [0010] with the following amended paragraph:

[0010] Figure 4 illustrates decoding an instruction in a first mode; ~~and~~

Please replace paragraph [0011] with the following amended paragraph:

[0011] Figure 5 illustrates decoding a system command from a second instruction set in a first mode**[[.]]:**

Please add the following new paragraph [0011.1]:

[0011.1] Figure 6A illustrates a method in accordance with at least some embodiments; and

Please add the following new paragraph [0011.2]:

[0011.2] Figure 6B illustrates a method in accordance with alternative embodiments.

Please add the following new paragraph [0031.1]:

[0031.1] Figure 6A illustrates a method in accordance with at least some embodiments. The method starts (block 600) and moves to decoding the current bytecode (block 602). Concurrently with decoding the current bytecode, the succeeding bytecode is pre-decoded (block 604). Thereafter, a determination is made as to whether the pre-decoded bytecode is a predetermined prefix (block 606). If so, decoding of the predetermined prefix is skipped (block 608), and the method ends (block 610). If the succeeding bytecode is not a predetermined prefix (again block 608), the process ends (block 610).

Please add the following new paragraph [0031.2]:

[0031.2] Figure 6B illustrates a method in accordance with alternative embodiments. The method starts (block 620) and moves to decoding the current bytecode (block 622). Thereafter, the succeeding bytecode, which may be the predetermined prefix discussed above, is detected (block 624). Finally, the process ends (block 626).